ABBOTTABAD UNIVERSITY OF SCINECE AND TECHNOLOGY :

:SYSTEM REQUIREMENT SPECIFICATION DOCUMENT:

:NAME: MUHAMMAD ABDULLAH QAZI :

:ROLL NUMBER : 14984 :

:PROGRAM : BSSE :

:SEMESTER : 4 {B}:

:SUBJECT : SODTWARE CONSTRUCTION AND DEVELOPMENT:

:SESSION : SPRING 2025 :

:SUBMITTED TO : MAM SAMMAN SHAHEEN:

1. Introduction

1.1 Purpose

This document defines the functional and non-functional requirements for the CLI-Based Task Manager System developed in C++.

1.2 Scope

This is a command-line application that enables users to manage tasks with features like add, view, update, delete, mark as complete/incomplete, and filtering/search. Tasks are stored in files.

1.3 Intended Audience

- Course Instructor
- Project Evaluator
- Future Developers
- Students for Learning Purposes

1.4 Intended Use

Designed for educational use, showcasing modular development, file handling, and object-oriented programming in C++.

1.5 Definitions and Acronyms

- **CLI**: Command Line Interface
- **CRUD**: Create, Read, Update, Delete
- **SCD**: Software Construction and Development

---

2. Overall Description

2.1 Product Perspective

This is a standalone system developed in C++ using standard libraries and modular design.

2.2 Product Functions

- Add/View/Update/Delete tasks
- Search tasks by ID or keyword
- Filter by status
- Save/load tasks using file I/O

2.3 User Characteristics

Users should know how to operate a terminal and interact with CLI-based applications.

2.4 Constraints

- Single-user system
- Local storage
- No graphical interface or networking

2.5 Assumptions and Dependencies

- Valid inputs are provided

- C++ compiler is available
- File read/write permissions exist

---

3. Functional Requirements

FR1: Add Task
User can add a task with title, description, and due date.

FR2: View Tasks
Display all tasks with full details.

FR3: Update Task
Update a task by entering its ID.

FR4: Delete Task
Delete a task by entering its ID.

FR5: Mark Task
Mark task as complete or incomplete.

FR6: Filter Tasks
Filter tasks by status (completed/incomplete).

FR7: Search Tasks
Search by keyword or task ID.

FR8: Save/Load Tasks
Tasks are saved and loaded using file I/O.

---

4. Non-Functional Requirements

NFR1: Usability
Clear CLI interface using text menus.

NFR2: Reliability
Handles invalid input gracefully.

NFR3: Maintainability
Code is modular for easy updates.

NFR4: Portability
Runs on any OS with a standard C++ compiler.

5. System Modules

main.cpp	Program entry and CLI menu
task.hpp / task.cpp	Task class and methods
file_manager.cpp	Handles file input/output
date_util.cpp	Validates and formats dates
search.cpp	Searching tasks by ID or keyword
filter.cpp	Filters based on completion status
ui.cpp	CLI prompts, input handling
test.cpp	Testing the core features

---

6. User Interface

- CLI menu with numbered options
- Prompts and console feedback
- Input via `cin`, output via `cout`

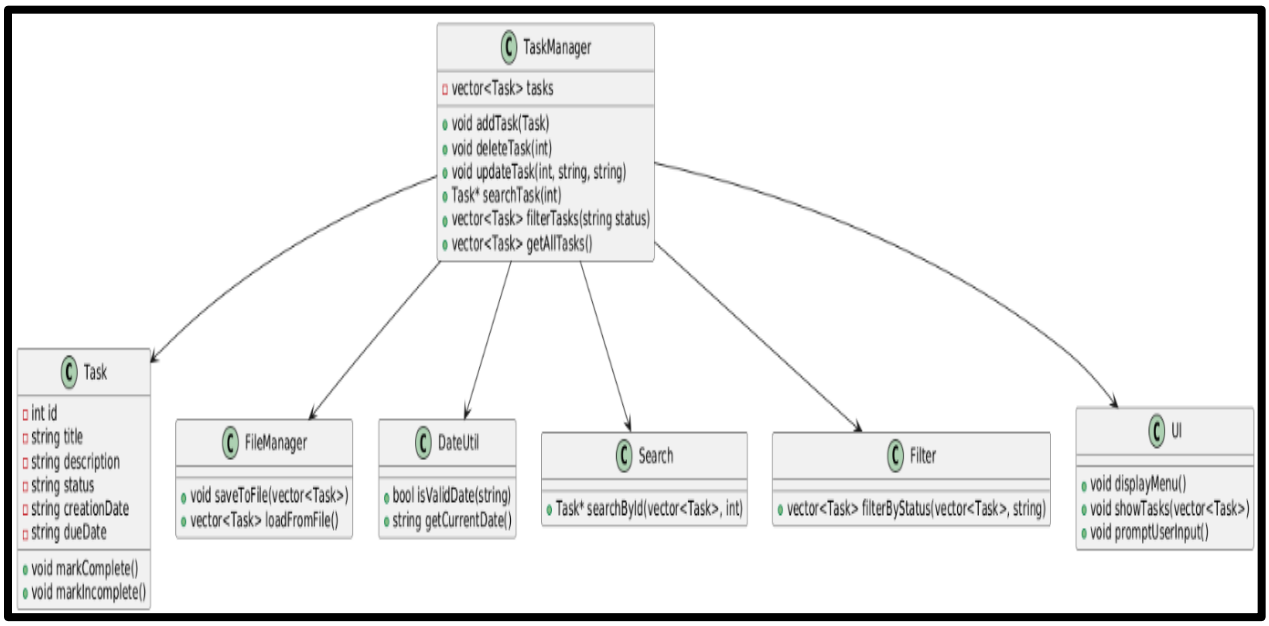
---

7. Future Enhancements

- Add GUI with Qt or another library
- Add task prioritization
- Multi-user functionality
- Add deadline reminders

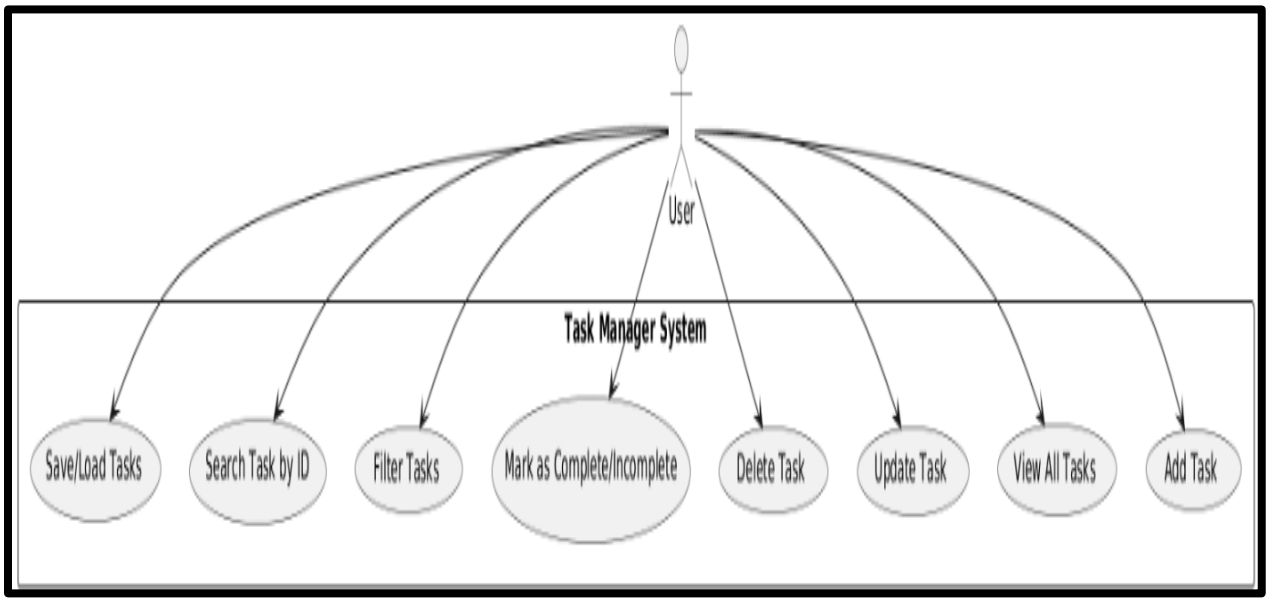
---

8. Class Diagram



---

9. Use Case Diagram



---

10. Appendix

10.1 Development Tools

- Language: C++
- IDE: Code::Blocks / VSCode
- Compiler: g++

10.2 File Structure

- `.cpp` and `.hpp` files in project root
- `tasks.txt` or equivalent for file storage
